Towards developing an intelligent graph restructuring algorithm for graph based stories using Google Maps

A Thesis
Presented to
The Academic Faculty

by

Sanjeet Hajarnis

In Partial Fulfillment
of the Requirements for the Degree
Bachelor's in Computer Science with Research Option in the
School of Computer Science

Georgia Institute of Technology
May 2010

Towards developing an intelligent graph restructuring algorithm for graph based stories using Google Maps

Approved By:

Dr. Mark O. Riedl, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Blair McIntyre
School of Computer Science
Georgia Institute of Technology

Dr. Ashwin Ram
School of Computer Science
Georgia Institute of Technology

Dr. Amy Bruckman
School of Computer Science
Georgia Institute of Technology

Date Approved: May 7th, 2010

Towards developing an intelligent graph restructuring algorithm for graph based stories using Google Maps

Abstract

Location-based games are gaining popularity because of their unique feature of having the players move in their environment. This feature serves dual purpose (i) of allowing the players to stay fit by walking outside while playing their location based game and (ii) of separating them from their routine sedentary work. Unfortunately, scalability becomes a huge concern for location based games because a particular (location-specific) game involves various landmarks and checkpoints in the vicinity of the current location. Thus, in order to cater to a sizeable audience and variety of locals, location based mobile games must be available for all or most playable locations. Having a database of games for all possible locations that players would wish to play in seems extremely infeasible and unreasonable. To overcome the afore-mentioned problem, this thesis proposes a solution that would attempt to eliminate pre-storing all the location specific games. The solution intelligently translates stories from their original locations into new locations by finding similarities between the locations. Thus, every time a user requests a game for a new location, an intelligent system restructures one of the previously written games for different locations to match and map to the new location the users intend to play in, via a web-based authoring tool. The translation algorithm restructures original story into new story using analogical reasoning, heuristic hill-climbing and dynamic programming. Analogical reasoning is performed with the location-specific information obtained from Google Maps. This thesis uses heuristic hill-climbing algorithm to obtain the optimal mapping between the original and new location by searching through the space of similar checkpoints/landmarks between the two locations and dynamic programming and memorization optimizes the algorithm's

performance by avoiding excessive re-computation. This thesis attempts to analyze the effects and success of using analogical reasoning with hill-climbing and dynamic programming in restructuring stories from original location to new location.

CHAPTER 1

INTRODUCTION

For the past three decades, computer games or console games have been developing with the advent of newer technology and innovative ideas like better graphics cards, optimized consoles etc. Unfortunately, the format of these computer games dictates the player to focus incessantly on their computer, television or console screens which can lead to players staying involved in the game in a sedentary manner. This reduces player's outdoor activities greatly. Wii has emerged as a solution which involves players to perform physical actions but it is possible avoid exertion while playing Wii games because they are predominantly indoors. One solution to the above games is to develop location-based games (LBG) that would enable the players to explore their environments while playing the games.

As LBG technologies improve, more people get interested in playing LBGs and remain fit by moving around in their surrounding environment. Players can play such location-based games using their wireless mobile devices while interacting with their peers, teammates or opponents in an urban environment. With LBGs "it is possible to create a wide variety of [game playing] experiences – both collaborative and competitive" [2]. Such collaborative and competitive environments can be created by developing games that require inter-player communication to achieve a common goal or a race towards a goal. This can also be achieved via a game that has a role-playing game flavor and involves interaction with non-playing characters (NPC) in the game.

Since LBGs are intertwined with the location they are built for, they pose a grave problem of scalability. A LBG must be present for every single location that user wants to play the game in and each game becomes location specific i.e., games developed for one location cannot be played in other locations. One solution to have games for myriad locations is to pre-develop games for large number of locations that player might want to play in. However, the afore-mentioned solution is extremely infeasible and unreasonable because it would be impossible to pre-develop such a large number of games and hence limits scalability of the games with respect to the locations games can be played in. This thesis proposes an approach to overcome this limitation with an intelligent web authoring tool that would allow players to play games built for other locations in any location that a player chooses to play in. Thus, the intelligent story translation algorithm eliminates the pre-computed games problem by allowing the users to translate games from one location to another.

This thesis involves building an intelligent algorithm for a web-authoring tool for LBG on GPS enabled mobile devices. The wireless devices require GPS because most of the game depends on the player's location. There needs to be a generic structure for LBGs thus every game is represented using a dependency graph. A dependency graph is a linked datastructure (Directed Acyclic Graph) of nodes or checkpoints from the stories which connects various checkpoints/landmarks in the story in a parent children relation. A parent-children relationship between the two checkpoints implies that parent checkpoint must be visited before checkpoint child. Hence using a dependency graph for a story ensures that the player takes the assumed path to reach the goal and complete the game. See Technical Background section for an illustration of a story in form of a dependency graph.

APPLICATION AND IMPORTANCE

Current video game innovations and investments are tending towards two major fields namely (i) geo-location involving interaction with environment and (ii) social networking involving interaction with friends playing similar games. Moreover, video games lifecycle requires constant human intervention for maintaining the games for various players and developing patches for old games. This structure requires an infrastructure that supports a huge player-base. This thesis proposes an infrastructure where the players would themselves provide creative games and all of the user-made games can be played nationwide once they are translated appropriately to the player's current location. Thus, using the authoring tool and translation algorithm would eliminate the requirement of maintaining large number of games. This maintenance involves storage of large number of games, developing new games periodically to avoid monotonous use of the same old games.

TECHNICAL BACKGROUND

Technical background for this thesis involves the following important concepts: (i) Dependency graphs, (ii) Analogical reasoning and (iii) Intelligent translation Algorithm

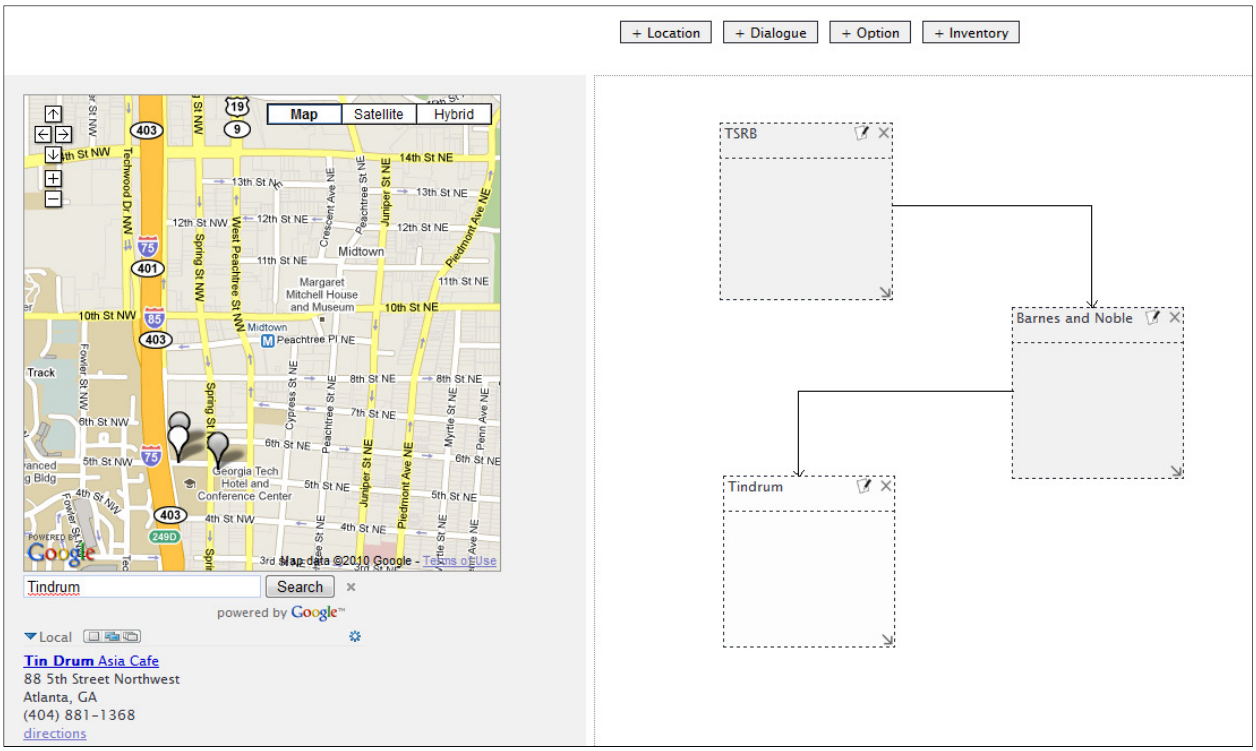

Figure 1
A dependency graph representative of a story that starts from location TSRB (a research building at Georgia Tech). The story is linear and follows the path TSRB to Barnes & Noble to Tindrum

In Figure 1, the story begins at TSRB (a research building at Georgia Institute of Technology) and the player would walk to TSRB. Once that checkpoint is clear, the player can move to Barnes & Noble (next checkpoint) and further to Tindrum. Thus, TSRB and Barnes & Noble are in a parent-child relationship where TSRB is the parent while Barnes & Noble is the child node and the player cannot complete the story unless he/she finishes the story in the order TSRB -> Barnes & Noble -> Tindrum. This generates a path a player should follow. Consider Figure 2 where the story also begins at TSRB but after finishing TSRB checkpoint, a player can either go to Tindrum or Barnes & Noble. Thus, a story not only has a linear path for the player but also a branching path for that player which would ensure that the story can be played more than once without performing the same actions repeatedly and keeps it interesting.

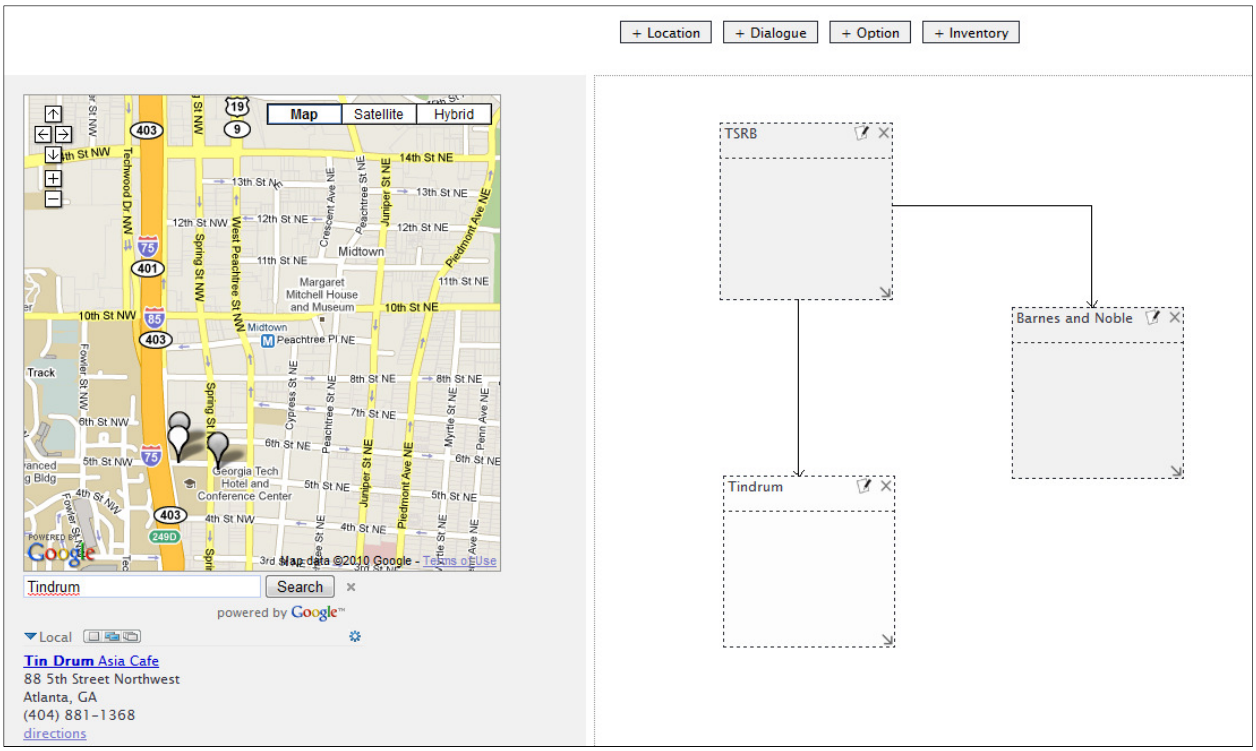

Figure 2
Figure 2 is also a dependency graph which represents a branching story which starts at TSRB and the player can either go to Tindrum of Barnes & Noble.

Since one cannot develop games for every single location, an authoring tool would allow the authors to use games developed in other locations and transform them into their current location and play it. The authoring tool's algorithm would implement analogical reasoning to restructure a game from one location to another by finding similarities between the buildings and places in the first location and another. Analogical reasoning relies on obtaining similarities between the two compared objects. Similarly, the algorithm analyses the similarities considering various facets of the two compared objects namely the semantics of the objects, the physical structure, objects behavior, effects caused by the objects.

While translating the story from original story to new story, these two important factors are considered: (i) semantics and (ii) distance. During translation, the authoring tool ensures that

the semantics of the original and new story are similar because the landmarks in original game should be consistent with the new story. The intuition behind this factor is that a particular location exists in the original story for a particular reason and the new story should have an appropriate location that matches the reason. For instance, a story could include TSRB as a location for research building or tall building or something different. This concept must be translated into the translated story. Moreover, the distance preservation is important because both the original and the new story should have about the same distance to travel while playing the game. Not only does the distance preserving feature promote consistency but it also promotes possibilities of have competitive games between people at different location. The above two translation factors ensure that the original game is translated to new story as the original stories author intended to be played. Translation of the games mainly relies on analogical reasoning.

In Figure 3, analogical reasoning is depicted between planet-electron pair and sun-nucleus pair by comparing how planets and electrons share common properties of attraction and revolution in an orbit about sun or nucleus respectively. There are two well-known analogy - finding algorithms namely Structure Mapping Engine (SME) and Connectionist Analogy Builder (CAB). SME models some aspects of human analogical processing. It computes an approximate "best" interpretation of a sentence and also generates alternate interpretations when required. It focuses on "the mapping [of concepts in sentences] to produce relevant yet novel, inferences." [6] CAB model attempts to emulate human cognition. It "determines which elements of two representations play compatible roles and places them into correspondence." [7] This model provides compatible elements in two different representations efficiently. Narrative story generation uses analogical reasoning to transform an existing story in a specific domain of concepts (source story) into new domain of concepts (target story) [8].

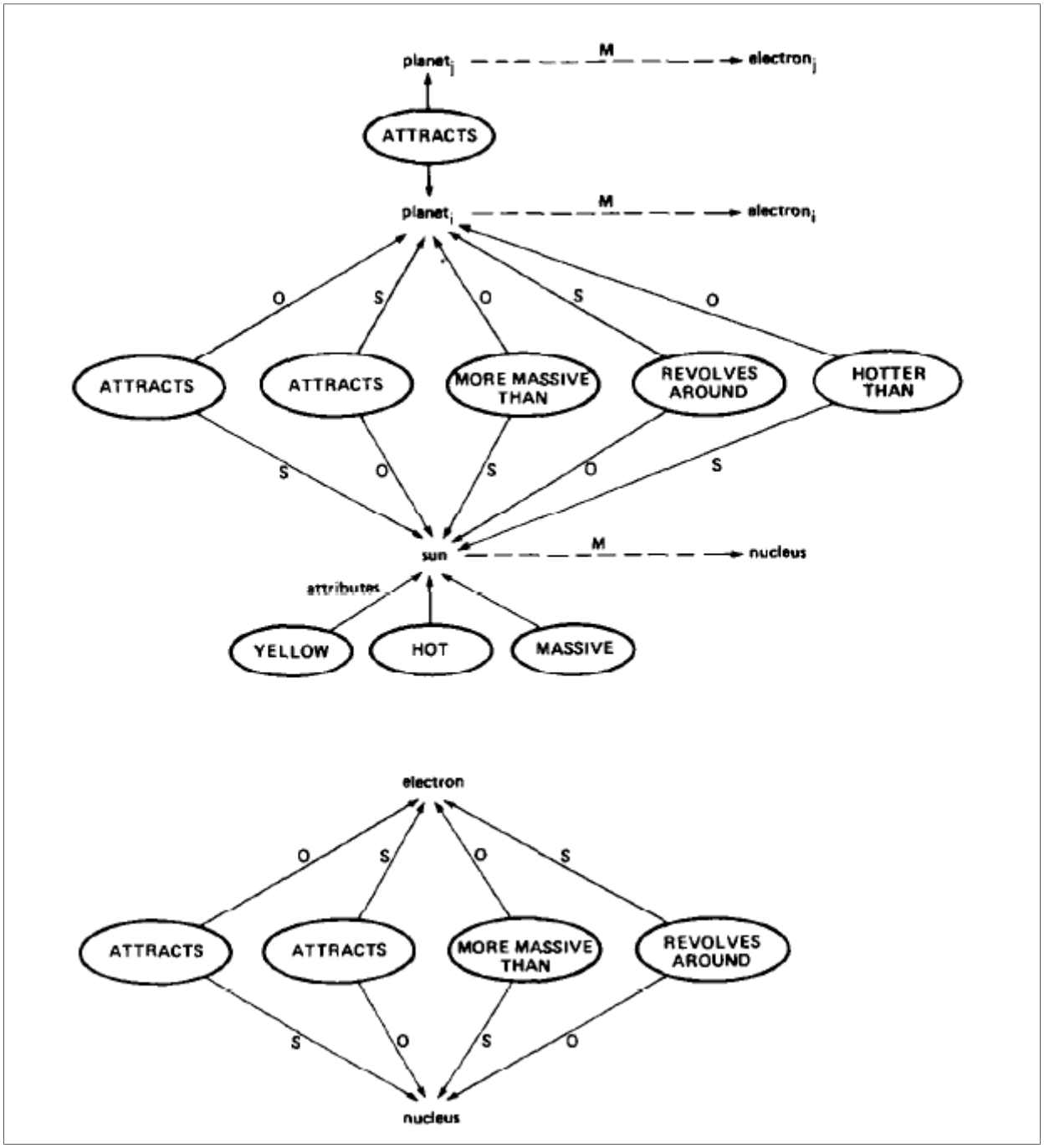

Figure 3

Generating such analogies in location-based games, stories or concepts requires information processing and retrieval. The problem of Information Retrieval (IR) has a few solutions that involve using algorithms that "perform fully automatic indexing and retrieval of

text" [3] or any other form of information available. Some of the most used IR algorithms use indexing and clustering of the documents [12]. Such IR systems overcome the problems posed by limitations of mobile devices and retrieves relevant information like user's location [10]. Moreover, such system can be used to develop accurate tag suggestions for various user developed contents [9].

The analogical reasoning method used in this thesis shares similarity with CAB and SME while attempting to find similar landmarks in the translated game. In this thesis, model for information retrieval, however, is quite different. In this work, Analogical Reasoning and Information retrieval are predominantly done using information from Google Maps. The data retrieval from Google Maps includes information about the location of various businesses, buildings and other important locations. The translation algorithm performs data retrieval from Google Maps based on user-generated tags associated with locations and landmarks. These tags provide generic information about the locations and hence help in finding similar landmarks in new location. For instance, a location marker on Barnes & Noble would include tags like bookstore. These tags are extremely helpful in translating the original locations into new locations. (Here translation means the mapping between the original checkpoints and new checkpoints) Currently, all the tags are user-generated but in the future, tags can be stored in a hierarchical tree structure which could further help the algorithm select the semantically closest translation. For instance, if Tindrum, a Thai restaurant on the Georgia Tech campus, is one of the original checkpoints, it is extremely difficult to find another Tindrum in the vicinity of a new location. Instead Tindrum can be stored into the hierarchy as follows:

Restaurants → Asian Cuisine → Thai Cuisine → Tindrum

Thus, in case of lack of perfect match on Tindrum, the authoring tool can move up the hierarchy to find nearby Thai cuisine restaurants and keep moving upwards in the hierarchy until a suitable match is obtained. In case of complete failure (i.e., authoring tool is unable to find a match) the tool displays an empty checkpoint and notifies the user to fill the location appropriately to suit the game storyline. Thus, the author of the new game would now have an option of continuing with the new structure or modify it and play on player's device.

Examples of translations from original location to new location:

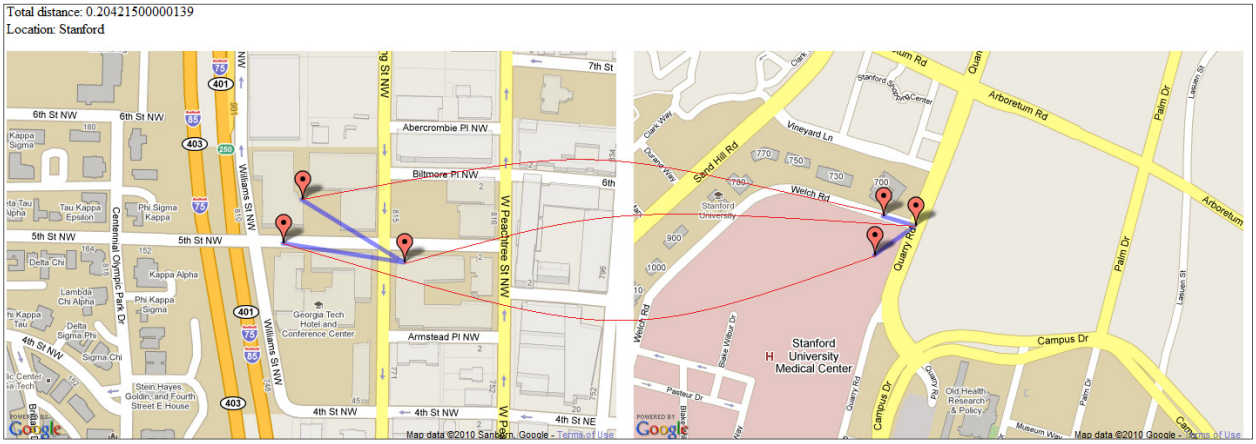

Figure 4

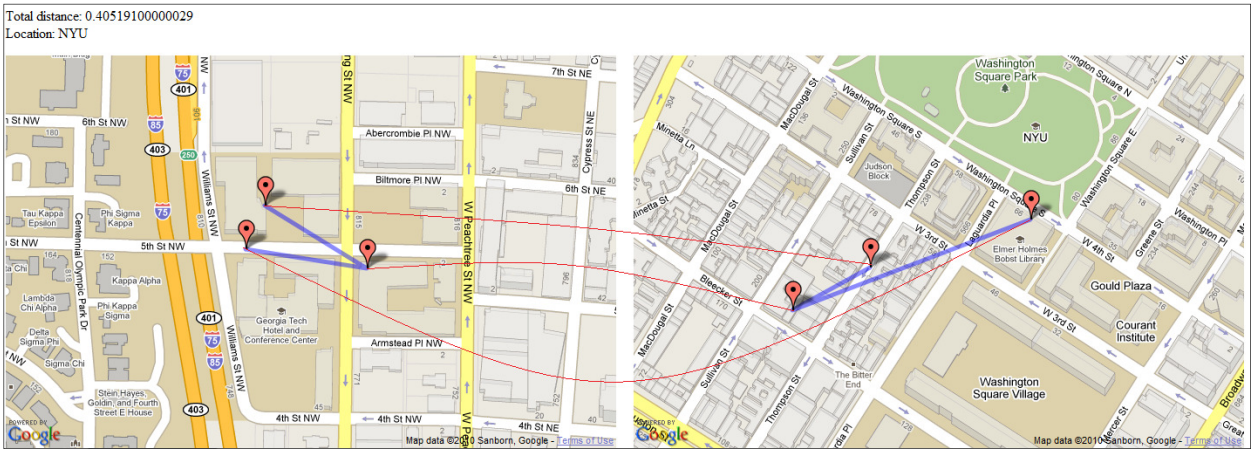

Figure 5

Figure 4 depicts the translation of story on Georgia Tech campus to its analogous locations on the Stanford campus while Figure 5 depicts the translation of story on Georgia Tech campus to its analogous locations on the NYU campus.

CHAPTER 2

PROBLEM DESCRIPTION AND SOLUTION

This section focuses on the translation algorithm and challenges that occurred while developing one. The thesis would discuss the problem translation algorithm attempts to solve followed by our approach to reach the solution.

The tags corresponding to each location are entered into Google Maps which performs analogical reasoning. Google Maps outputs analogous locations between the original location and the new location. After obtaining the information, the next challenge involves developing an algorithm that would intake the above information and output an optimal path from the first location in the game to the last location. In this problem, the optimal story path can be defined by its similarity to the original path in terms of (i) the distance between the two checkpoints (ii) angles formed by path in the checkpoint. To formalize the problem at hand let us consider that there are L locations in the original game. As we retrieve information from Google Maps, it returns N analogous locations for each location in the original game.

2.1 Initial Approach

The above problem can be viewed as a simple state-space search problem. We decided to perform an exhaustive search on the data obtained from Google Maps and obtain all possible paths from start location to end location. All these paths would be ranked using an appropriate heuristic. The initial idea of heuristic was to match the total distance of the path to the original path such that if the games were played in a competitive format in future, they would be fair because every player walks the same distance and having the same distance would keep the objective of the new game consistent with the original game. This approach is guaranteed to give us the most optimal solution with respect to our heuristic, but it has many limitations. Google Maps returns many locations that are analogous to the queried old locations and that are under some walkable distance from the previous checkpoint. Moreover, an exhaustive search of the state space would require a scan through all possible path combinations. The running time for this algorithm would be $O(N^L)$ in the worst case scenario because Google Maps would not return the same number of analogous locations. If a particular story contains large number of locations or large number of analogous locations, the translation would be time-intensive. These limitations in the algorithm required optimization and change in heuristic.

2.2 Dynamic Programming

An extensive search through the state-space of paths performed repetitive computations. We decided to use Dynamic Programming with memoization to avoid the repeated calculations. Dynamic programming (DP) is generally applied to optimization problems. In the case of dynamic programming, there are multiple solutions that evaluate to a certain value and DP's goal is to obtain an optimal solution. "The development of dynamic programming algorithm can be broken into a sequence of four steps: (i) Characterize the structure of an optimal solution (ii) Recursively define the value of an optimal solution (iii) Compute the value of an optimal

solution in a bottom-up fashion. (iv) Construct an optimal solution from computed information" [13]. A typical dynamic programming solution is to solve the Edit-distance problem. In an Edit-distance problem, we are given two words (source and target) and we must calculate an optimal way to transform the source word into a target word given the actions copy, delete or insert. Different variations of the Edit-distance problem assign different point values for actions like copy, delete and insert but for the purpose of this example we will assume that delete and insert cost 1 unit while copy costs 0.

Consider the words fast (source) and fat (target). An optimal edit distance for this problem can be computed using dynamic programming. A dynamic programming solution saves the subproblem's optimal solution in the table. Consider Table 1 as reference for this discussion. In Table 1, the source sequence is represented as columns while the target sequence is represented as rows. The easiest subproblem in this case is to transform empty into empty where the edit-distance would be 0 since no action is performed.

		F	A	S	T
	0	1 (Delete F)	2 (Delete A)	3 (Delete S)	4 (Delete T)
F	1 (Insert F)	0 (Copy F)	1 (Delete A)	2 (Delete S)	3 (Delete T)
A	2 (Insert A)	1 (Insert A)	0 (Copy A)	1 (Delete S)	2 (Delete T)
T	3 (Insert T)	2 (Insert T)	1 (Insert T)	1 (Delete S)	0 (Copy T)

Table 1 Dynamic Programming matrix for edit-distance problem

In table 1, each cell outputs an optimal solution to transform the source at that point into target at that point. For instance, consider the cell in blue circle. Here the edit distance for transforming F (source) into FA (target) is 1 by taking the following actions. Copy F and Insert

A. Similarly to find an optimal distance for transforming the entire source into target we consider the cell entry circled in red. The edit distance of the entire transformation is 1 and it is done by undergoing the following actions. If we perform the following actions: copy F, copy A, delete S, copy T we transform “fast” into “fat”. Thus it can be observed that Dynamic Programming seems like a plausible solution for problems similar to edit-distance problem.

The translation problem is also similar to any typical dynamic programming problem because the translation requires an optimal solution from a start location to the end location. It fits the afore-mentioned four characteristics of a typical dynamic programming problem. An optimal solution would be an optimal sequence obtained from the dynamic programming matrix. In order to get optimal solution between first and last checkpoint, the solutions for other checkpoints between first and last must be optimal.

For instance, once the best location is selected as the first location, it is stored for future reference and there is no need to recompute the best location. Using Dynamic Programming for this problem is intuitive because the problem can be easily broken down into smaller overlapping subproblems and the solutions to these subproblems can be stored to reuse in later computations. This problem can either be solved from start to end or end to start (i.e., the dynamic programming matrix can be built by either considering the first location first or last location first because they an optimal path would be the same). Our approach involves calculating an optimal path from the first location the last location. For instance, consider a story with 5 locations A-E. The best solution from A to E can be described as the best solution from A to D and D to E. Similarly to calculate best solution from A to D would be a combination of best solution from A to C and C to D. It is easy to notice a recurrence relation in the above algorithm such that certain computed results can be reused and this algorithm would save repetitive computation from the

initial approach. The translation algorithm relies on the concept of finding best solution between two checkpoints in the story. A best solution is a solution which is the most optimal solution considering the heuristic used. If the recurrence relation for our dynamic programming solution is represented as R[n] where n is the last location in the story then,

$$R[0] = 0, R[1] = best\ solution\ for\ 0th\ and\ 1st\ locations\ and\ R[n] = R[n-1] + best\ solution\ for\ (n-1)st\ and\ nth\ location$$

Although there is a striking resemblance between the two problems, translation problem has a caveat. The story paths are not always linear instead they could be branched in a complicated manner. The above mentioned dynamic programming solution works on a inherent assumption of linear progression (e.g., 1 followed by 2, 2 followed by 3 … n followed by n-1) This linearity makes the recurrence relation easier to compute by an iteration. In the case of branching stories the progression would look something similar to 1 followed by 2, 2 followed by 3 and 4, 3 followed by 5, 4 followed by 6 and 7 … and not linear. Obtain a recurrence relationship for a branching story requires information about previous locations.

In order to overcome this limitation, our solution for translation includes an additional datastructure apart from the dynamic programming table to store information about a locations' parent (or previous location). Having this information allows us to find optimal solutions for stories that are not necessarily linear. Consider the following pseudo code which incorporates parent information with dynamic programming to calculate optimal solutions for branching stories.

```
locationArray[][]
min ← large number
edges[]
```

```
for i ← to size of locationArray[0]
        bestLocation0 ← null
        bestLocation1 ← null
        for j ← to size of locationArray[1]
                dist ← compute distance between locationArray[0][i] and locationArray[0][j]
                 if abs(dist – edges[0]) < abs(min-edges[0]) then
                        min ← dist
                        bestLocation0 ← locationArray[0][i]
                        bestLocation1 ← locationArray[0][j]
bestLocation[0] ← bestLocation0
bestLocation[1] ← bestLocation1
bestDistance [0] ←min

for i ← 2  to size of locationArray
        min ← large number
        tempLocation ← null
        for j ← 0 to size of locationArray
                dist ← distance between current location and its parent location
                if abs(dist – edges[i-1]) < abs(min-edges[i-1]) then
                        min ← dist
                        tempLocation ← current location
        bestLocation[i] ← tempLocation
        bestDistance[i] ← bestDistance[i-1] + dist
```

After the computation, bestLocation array would have an optimal path while bestDistance[size of locationArray – 1] would have the distance of an optimal path. The above mentioned pseudo code performs with the heuristic preferring the closest distance to the original story. Refer to the next section for discussion on the heuristic.

2.3 Heuristic

The initial heuristic focused on primarily on having the same total distance for both original story and translated story. Hence the heuristic was calculated after all the paths were found. Since the optimized algorithm checked for the best solution between two checkpoints, the heuristic needed to have a local factor where goodness of a path would be decided just by the distance between two checkpoints. In order to get this feature, the heuristic decided to match the inter-checkpoint distance of both original and translated story. This change in heuristic not only

solved the problem of having inter-checkpoint consistency between original and translated story but also ensured that the total distance between the two stories stayed consistent. If every inter-checkpoint distance was consistent within some error ε, then the total distance would be within L* ε error. Consider O_{ij} and T_{ij} be the edge distance between location i and j in original and translated story.

$$\forall Oij, \exists\ Tij\ such\ that\ |Tij - Oij| < \varepsilon ij\ then\ |\sum(Tij) - \sum(Oij)| < L * \varepsilon max$$

$$where\ where\ \varepsilon ij > 0, \varepsilon max\ is\ the\ max\ error\ in\ \varepsilon ij\ and\ L\ is\ the\ number\ of\ locations.$$

For the heuristic to obtain optimal inter-checkpoints distance, there has to be a compromise on the consistency with total distance but the tradeoff can be bounded by the number of check-points and the maximum inter-checkpoint error. The objective of the dynamic programming algorithm would be to minimize this inter-checkpoint error hence the heuristic and dynamic programming combined together can put the total error bounded and low.

We decided to add a few additional features to the heuristic and the algorithm to provide better results after translation. The heuristic also checked for the angles between three check-points by taking considering the latitude and longitudes of each locations are x and y co-ordinates in a Euclidean 2D space. Here we assume that earth is a 2D plane but this assumption seems fair considering the proximity of these locations amongst each other. The motivation behind this heuristic was that the player would be extremely annoyed if he/she would have to move back and forth on the map to complete the quest. In order to avoid such zigzag path formation, the heuristic favors larger angles between the three checkpoints. Acute angles tend to generate zigzag paths hence they should be avoided. Unfortunately banning all acute angles would be detrimental for the algorithm because there could be stories that have a cycle which

could require acute angles. Thus, a perfect balance must be found in the heuristic for inter-checkpoint distance and angle between three checkpoints.

CHAPTER 3

EXPERIMENTAL DESIGN, RESULTS AND DISCUSSION

The translation algorithm must be evaluated for correctness. The two major evaluations criteria are (i) Distance preservation and (ii) Semantic preservation. By performing series of experiments, we can measure the algorithm's performance based on the structure of the translation graph and how similar it is to the original story. The series of experiments involve performing translation of stories from Georgia Tech to five other Universities in the United States. These experiments would help us evaluate the accuracy of the translation based on how similar is the translated story to the original story. This evaluation can be done by visual inspection. The second experiment would evaluate the distance consistency for the five universities. Translated story's distance should be close to the original story.

3.1 Translation Experiment

The translation experiment evaluates the success of the translation process based on how well the stories have been translated from one location to another. The following figures show the translation of stories from Georgia Institute of Technology to another university. The original story at Georgia Tech requires the following sequence of action. (See Figure 6 for visual representation of the original story at Georgia Institute of Technology).

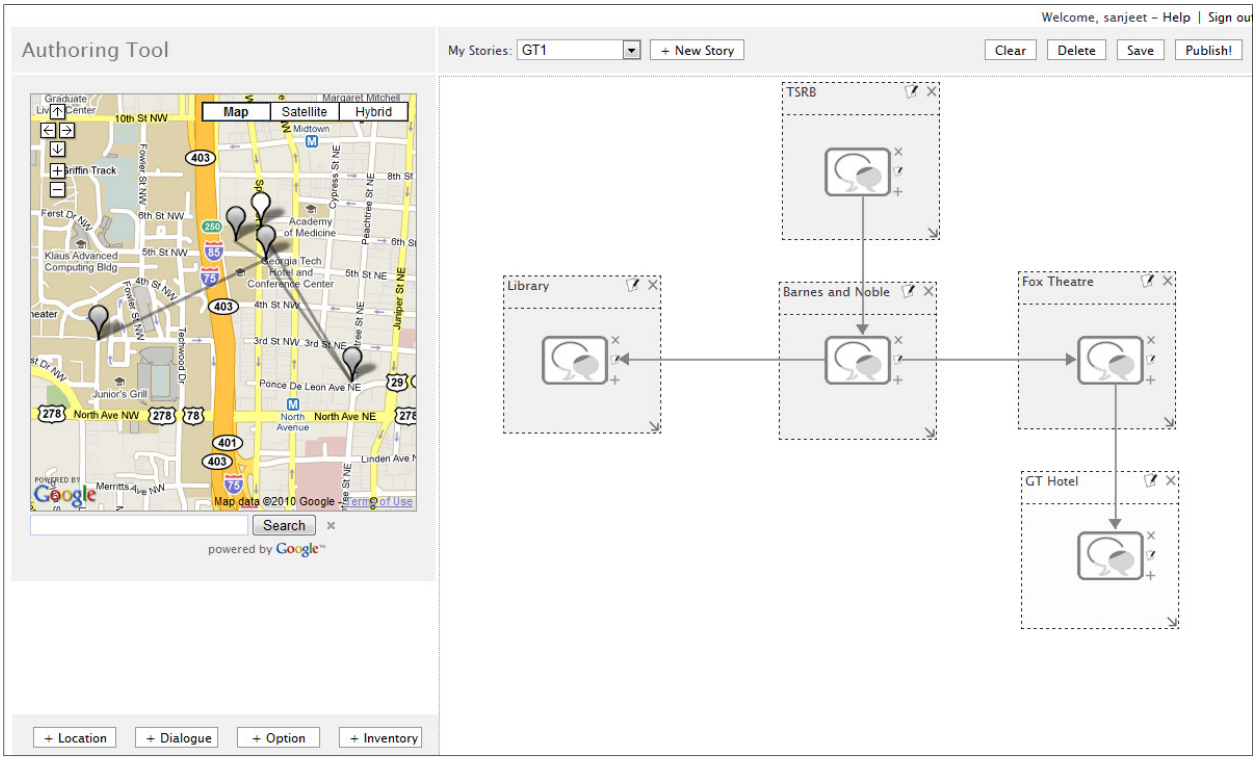

Figure 6 – Visual representation of the story based at Georgia Institute of Technology and used in the experiments

The story requires the players to complete their checkpoints in the following order. The story begins at Tech Square Research Building, followed by a visit to Barnes & Noble bookstore. The players has two options at this checkpoint, they can either go to the Library or Fox Theatre. In order to complete the story, the player must go from Fox Theatre to the Georgia Tech Hotel. The above mentioned story is used as an original story and then later translated for different universities.

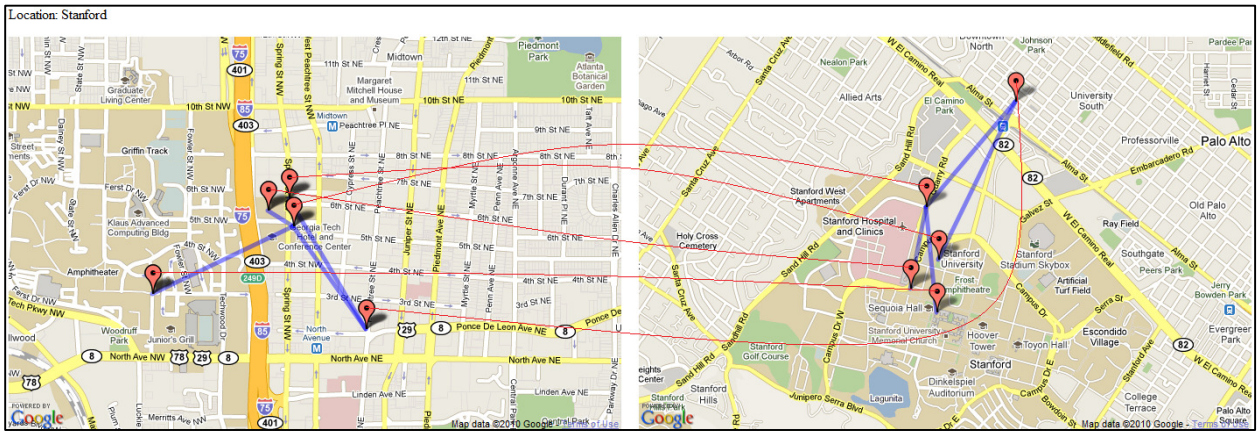

Figure 7 – Translation from Georgia Tech to Stanford University

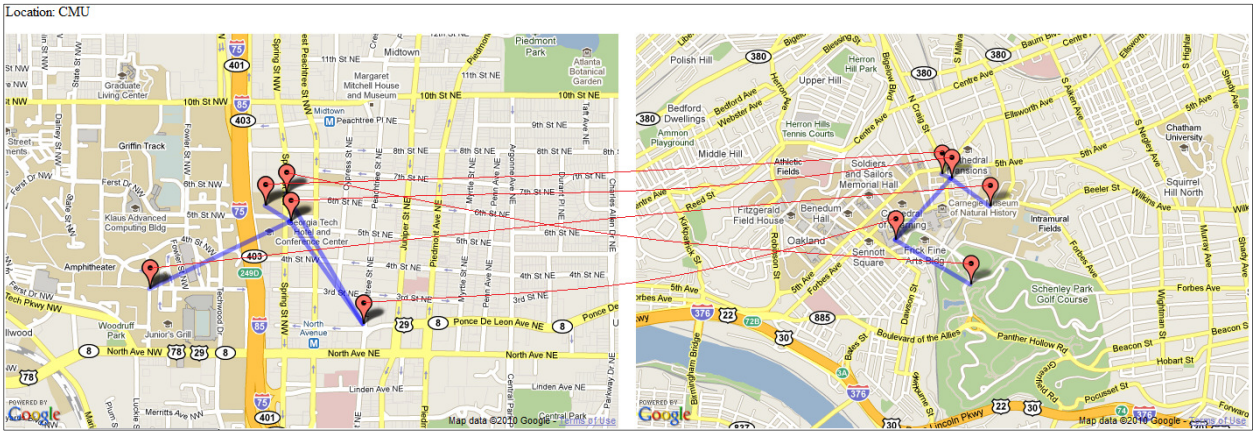

Figure 8 – Translation from Georgia Tech to Carnegie Mellon University

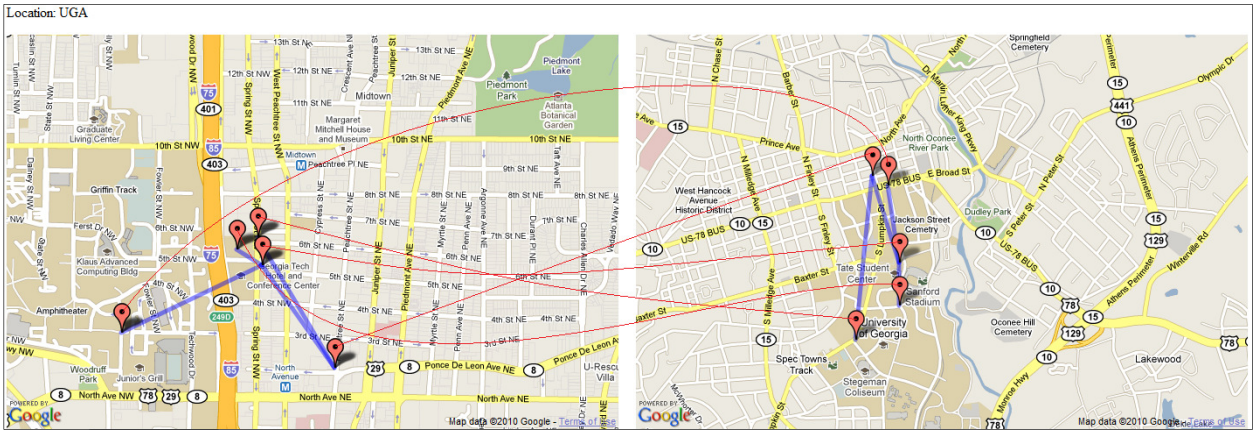

Figure 9 – Translation from Georgia Tech to University of Georgia

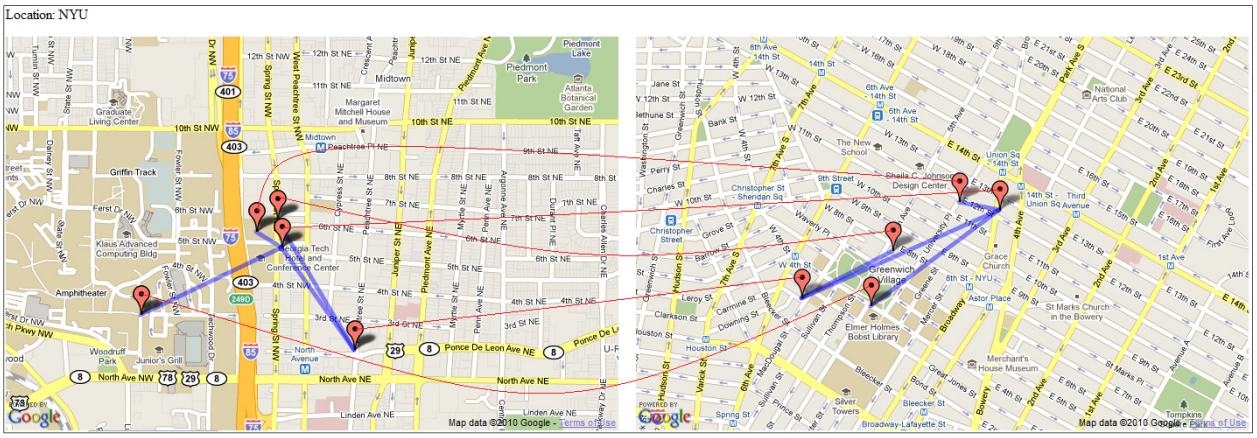

Figure 10 – Translation from Georgia Tech to New York University

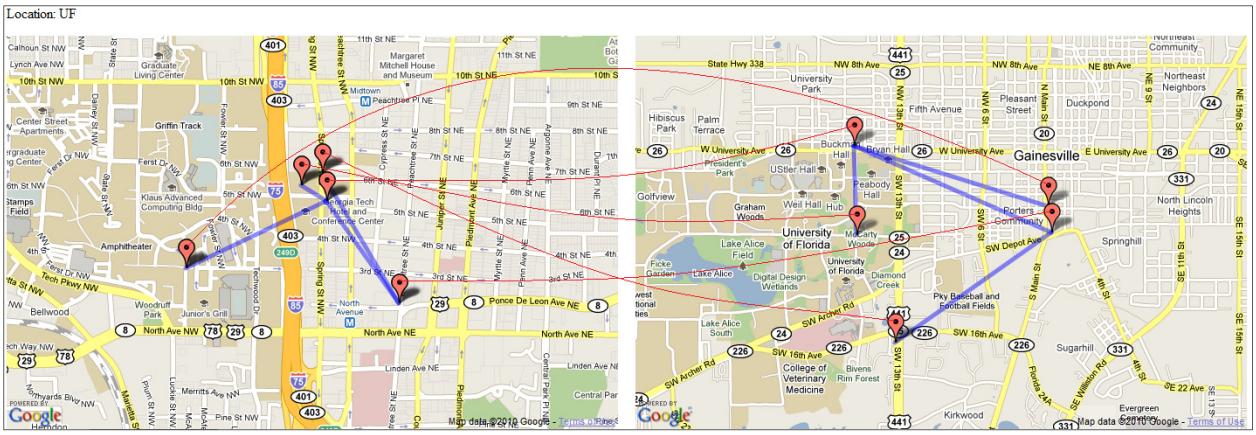

Figure 11 – Translation from Georgia Tech to University of Florida

3.2 Distance Experiment

The distance experiment relies on the closeness of inter-location distance between the original and the translated game. Table 2 details the inter-location distances for the original story at Georgia Institute of Technology. The inter-location distances are labeled as 1-4 where the 1st inter-location distance is the distance between the first two locations and similarly for 2-4.

Distance 0	Distance 1	Distance 2	Distance 3	Total distance
0.150258	0.608695	0.776807	0.85557	2.39133

Table 2

The objective of the translation algorithm is to obtain a path that is similar to the original story. Thus, the algorithm tries to keep the inter-location distance consistent. Following are the dynamic programming matrices that found optimal solutions for translations in various different universities. The dynamic programming solution matrix also follows the same naming convention as Table 2. Here the inter-location distances are labeled 1-4 as columns of the table and the rows of the table are search results of locations that are analogous to the original locations and hence potential locations to be included in an optimal solution.

Stanford	Distance 1	Distance 2	Distance 3	Distance 4

Result 1	0.55166	0.18249	0.29791	2.32102
Result 2	1.18405	0.79829	1.47288	0.33077
Result 3	1.00828	1.34055	1.00235	0.40409
Result 4	1.47754	0.15018	1.47754	1.15087

Table 3 – DP matrix for Stanford. Total distance: 2.756389

CMU	Distance 1	Distance 2	Distance 3	Distance 4
Result 1	0.86268	0.63927	0.31387	0.40361
Result 2	0.20153	1.72615	0.11297	3.22209
Result 3	0.19015	2.71973	0.46096	1.94869
Result 4	0.15176	1.08601	0.18581	1.63806

Table 4 – DP matrix for Carnegie Mellon University. Total distance: 1.655603

UGA	Distance 1	Distance 2	Distance 3	Distance 4
Result 1	0.93165	0.76717	0.09741	1.17908
Result 2	0.28053	0.78439	1.01728	0.33725
Result 3	0.78147	1.4694	0.88891	1.00235
Result 4	0.82128	1.0981	1.22553	0.4108

Table 5 – DP matrix for University of Georgia. Total distance: 2.938958

UF	Distance 1	Distance 2	Distance 3	Distance 4
Result 1	0.74702	0.62386	0.30075	2.14311
Result 2	0.6207	1.80649	1.86582	2.31257
Result 3	1.25231	0.92928	0.76006	0.72964
Result 4	1.41331	1.26297	1.05931	2.41874

Table 6 – DP matrix for University of Florida. Total distance: 2.734269

In Table 3-6 we can observe that the translation algorithm chooses an optimal path where the inter-location distances stay consistent. The arrows suggest the story path. Thus, the story would involve walks the highlighted inter-location distance to move from one location to another. Table 7 contrasts the total distance a player needs to travel in the original story at Georgia Tech and the translated story.

Translated Location	**Translated Distance**	**Original Distance**
Stanford	2.7564	2.3913
CMU	1.6556	2.3913
UGA	2.9389	2.3913
NYU	2.2622	2.3913

UF	2.7347	2.3913

Table 7

CHAPTER 4

CONCLUSION

Although this research is not complete, it possesses the potential of solving the scalability problem for the LBG. The intelligent authoring tool algorithm has shown that it is possible to obtain a translated story from original location to the new location by using Google Maps to perform analogical reasoning and information retrieval. Although there are instances where the translation algorithm does not find a perfect translated story, it shows reasonable success for the university shown in the above experiment.

The future work section discusses ideas on how to eliminate some of the limitations in the intelligent translation algorithm.

CHAPTER 5

FUTURE WORK

There are instances when the translation algorithm cannot find analogous checkpoints for the new locations. One of the reasons for this limitation is lack of relevant keywords. For instance, if one of the checkpoints in the original story was Tindrum Asia café then it might be possible that the new location does not have a Tindrum Asia café. One solution to this problem is to have a deontological hierarchy for each location which would ensure that the algorithm always finds an analogous checkpoint. For instance, the hierarchy for Tindrum Asia café would be Building → Restaurant → Asian Cuisine → Thai food → Tindrum Asia café. This hierarchy would allow the algorithm to look for Thai food if it cannot find Tindrum Asia café. Thus,

ideally the algorithm should obtain the analogy at the deepest level and continue backtracking until an analogy is found. Adding the above feature would make the algorithm robust.

References

[1] Benford, S., Flintham, M., Drozd, A., Anastasi, R., Rowland, D., Tandavanitj, N., Adams, M., Row-Farr, J., Oldroyd, A. and Sutton, J. Uncle Roy All Around You: Implicating the City in a Location-Based Performance. In Proc. Advances in Computer Entertainment (ACE 2004), ACM Press.

[2] S. Benford, R. Anastasi, M. Flintham, A. Drozd, A.Crabtree, C. Greenhalgh, N. Tandavanitj, M. Adams, and J. Row-Farr. Coping with uncertainty in a location based game. IEEE Pervasive Computing, 2(3):34-41, July-September 2003.

[3] Budzik, J., Hammond, K., and Birnbaum, L.2001. Information access in context. Knowledge Based Systems. 14.

[4] W. Chen, Y. Hayashi, L. Jin, M. Ikeda & R. Mizoguchi, An Ontology-based Intelligent Authoring Tool: *Proc. ICCE'98*, Beijing, China, 1998, 41-49.

[5] Devedzic, Vladan. (2004). Education and the Semantic Web. International Journal of Artificial Intlelligence in Education 14, 39-65.

[6] Forbus, K.D. and Oblinger, D. (1990). Making SME greedy and pragmatic. Proceedings of the Twelfth Annual Conference of the Cognitive Science Society.

[7] Larkey, Levi B., and Bradley C. Love (2003). CAB: Connectionist Analogy Buider. Cognitive Science 27, 781-94

[8] Riedl, M. and León, C. Generating Story Analogues. In Proc. AIIDE09: the 5th Conf. on Artificial Intelligence for Interactive Digital Entertainment, AAAI Press (2009), Palo Alto, CA, USA.

[9] S. Sood, K. Hammond, S. Owsley, and L. Birnbaum. TagAssist: Automatic Tag Suggestion for Blog Posts. ICWSM'07.

[10] Sood S., Kristian J. Hammond, and Larry Birnbaum. Low-fidelity location based information systems. In Nuno Jardim Nunes and Charles Rich, editors, Proceedings of IUI 04, New York, NY, USA, 2004.

[11] Vogiazou, Yanna, Bas Raijmakers, Erik Geelhoed, Josephine Reid, and Mark Eisentadt. (2007). Design of emergence: experiments with mixed reality urban playground game. Personal and Ubiquitous Computing 11, 45-58.

[12] Grossman, David A., and Ophir Frieder. Information Retrieval: Algorithms and Heuristics. Boston [u.a.]: Kluwer, 2002. Print.

[13] Thomas H. Cormen, Charles Eric Leiseron, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms. Chapter 15 Dynamic Programming. MIT Press – Cambridge (Mass.), 2009. Print.